Estimating Smart Contracts Performance

Nicola Elia[†], Francesco Barchi[†], Alessia Pisu[*], Livio Pompianu[*], Andrea Acquaviva[†]

***Abstract*—The employment of smart contracts to enable the execution of industrial distributed applications is continuously growing. Notably, the performance of these applications is tightly related to the execution performance of the underlying smart contracts operating on the target blockchain network. This paper delves into the impact of real-world environment on smart contract performance, presenting a holistic step-by-step methodology that allows the user to simulate a production-grade environment for deploying and benchmarking a smart contract. The methodology presented in this work has been applied to an industrially relevant case study, namely a IoT Structural Health Monitoring application. The results show how it enables organizations to make informed decisions regarding smart contract deployment and their scalability, by means of a quantification of the effects of constraints such as network limitations and blockchain configuration parameters.**

***Index Terms*—Blockchain, Industrial applications, Smart Contract, Performance evaluation, Benchmark**

I. Introduction

The growing adoption of blockchains across diverse sectors underscores the critical need for a thorough understanding of smart contract execution performance under the load conditions provided by real-world industrial scenarios. The significance of this kind of scenarios is highlighted when we consider the growing prevalence of blockchain-based infrastructures in critical or data-intensive industrial applications.

Smart contract execution performance is influenced by several interrelated factors: (I) The blockchain *internal components*, such as the consensus protocol. (II) Their *client implementations* that impact the execution speed of transactions. (III) The *computational capabilities* of the infrastructures, along with *resource allocation* for hosting blockchain nodes, directly impacting the efficiency of transaction processing. (IV) The *network infrastructure* that interconnects individual nodes within the blockchain network topology, having latency, throughput, and reliability properties that contribute to the overall responsiveness of transaction processing.

Taking into account these interconnected elements is fundamental for accurately estimating the execution performance of smart contracts within a blockchain.

In the context of distributed industrial applications, Average Transaction Latency (ATL) and Average Transaction Throughput (ATT) have been selected as primary Key Performance Indicators (KPIs) among the various metrics that can be considered when evaluating smart contracts.

In this paper, we introduce a methodology designed to assess the end-to-end performance of smart contracts within industrial scenarios independently of the underlying blockchain technology. Furthermore, we demonstrate the application of the proposed methodology on a real industrial use case of Structural Health Monitoring (SHM) [1]. For this, we develop a tool that automates the deployment of production-grade Hyperledger Fabric networks and facilitates the automated benchmarking of smart contracts, accounting for network conditions, blockchain configuration, and workload.

More specifically, our work makes two main contributions: *(I) Reference Methodology*: consisting of four phases, each with specific objectives, descriptions, and actions. *(II) Reference Implementation*: application of the proposed methodology to a real case study taken from literature.

II. Our proposal

In the context of smart contracts serving industrial applications, we introduce a four-step approach to evaluate and estimate the performance of production-grade deployments.

Firstly, the methodology begins with **Automated Blockchain Deployment**. It ensures repeatability and ease of experimentation by using automated procedures to deploy blockchain nodes. Key considerations include selecting the blockchain technology and its consensus protocol, configuring node parameters to mirror production environments, detailing the network's composition, and pre-loading the network to simulate the presence of other smart contracts. This aims to mimic real-world production-grade deployments closely.

Next, the focus shifts to **Emulating Real-World Network Properties**. Both permissionless and permissioned blockchains benefit from geo-distribution: the former because distributing consensus globally prevents attacks and avoids censorship or monopoly by any single entity or government [2]–[4], the latter because a geographically dispersed network is less susceptible to reliability issues, availability issues or single points of failure [5]–[8]. Network emulation allows for the adjustment of latency, jitter, throughput, and packet loss to replicate real-world conditions, ensuring that the system's performance can be accurately assessed, even in scenarios where the blockchain network does not yet exist.

The third step, **Smart Contract Deployment**, involves deploying the smart contract onto the network. This step is straightforward, requiring a sequence of blockchain-specific commands. It also includes selecting the type of transaction to target for benchmarking, with a preference for read-write transactions due to their higher computational demands.

Finally, the methodology leads to **Benchmarking**, where the smart contract's performance is assessed using two main key performance indicators (KPIs) that characterize the end-to-end performance: Average Transactions Latency (ATL) and Average Transaction Throughput (ATT). This step aims to understand how well the smart contract can meet the demands of its intended use case, and the potential room for optimizations.

[*]University of Cagliari, Via Ospedale 72, Cagliari, Italy
[†]University of Bologna, Viale del Risorgimento 2, Bologna, Italy
Corresponding author: pompianu.livio@unica.it

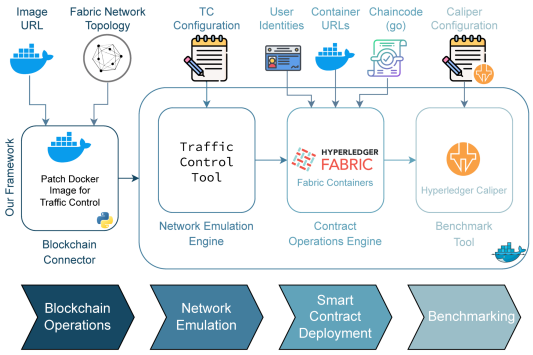

Fig. 1: Architecture of the reference implementation.

III. Case-study

We apply our methodology to a case study focused on a Structural Health Monitoring (SHM) application for a railway bridge [1], utilizing a blockchain-powered framework for secure data management via Hyperledger Fabric smart contracts.

Following the steps described in the proposed methodology, we develop a modular framework for setting up production-grade Hyperledger Fabric networks. For network emulation, we utilize the Traffic Control tool to simulate real-world network conditions such as latency and packet loss, based on network performance data from leading cloud providers to reflect realistic network conditions. The deployment process involves setting up the blockchain components as Docker containers, then instantiating the smart contract on the network, preparing for benchmarking. For benchmarking, we make use of Hyperledger Caliper, an open-source tool supported by the Linux Foundation, to measure the smart contract's performance focusing on average transactions latency and throughput. This case study not only demonstrates the application of our methodology but also provides a blueprint for future research, offering insights into deploying and evaluating blockchain-based applications in industrial settings.

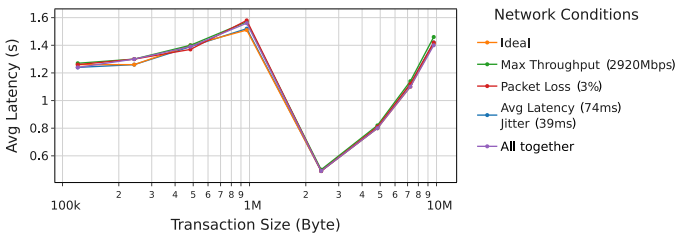

Fig. 2: Results of the first benchmark run in logarithmic scale.

IV. Experimental findings

Firstly, we evaluated the performance of the industrial application when subject to different environments, starting from ideal conditions and up to incorporating all the network constraints reproducing a real-world network. The results of this benchmark are depicted in Figure 2.

The real-world network conditions appear to have a negligible impact on the smart contract throughput with respect to an ideal deployment. However, we can observe that the resulting Average Transaction Latency (ATL) may be up to 6% higher than the ideal conditions.

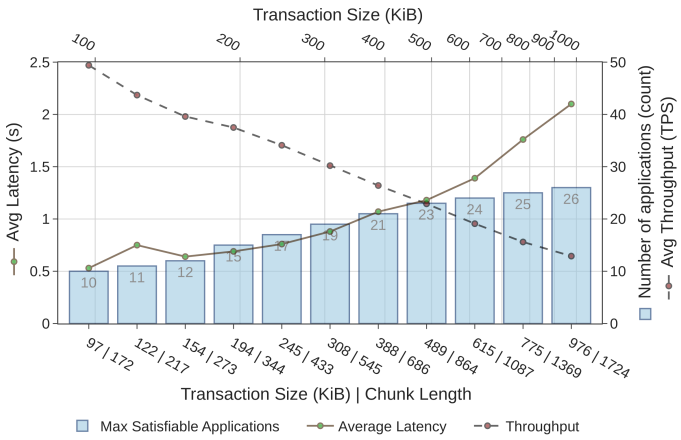

Fig. 3: The line plots show the maximum average latency and throughput that the system under test is able to provide when using different transaction sizes. The histogram shows the corresponding maximum number of running concurrent industrial applications. Logarithmic scale.

Secondly, we evaluated the capabilities of the application when stress-tested. For this purpose, we leveraged the fixed-load Caliper configuration, which varies the input TPS dynamically, ensuring that the backlog queue of undone transactions is always non-empty. The outcomes of this benchmark are presented in Figure 3. Considering each different transaction size configuration, we can measure the ATT and therefore we can compute the maximum number of satisfiable concurrent application as $N = \mathrm{TPS}_{\mathrm{Res}}/\mathrm{TPS}_{\mathrm{App}}$. We can notice how the ATT decreases as Transaction Size increases, whereas Average Latency demonstrates the opposite trend. More specifically, the Transaction Size can be adjusted to accommodate more concurrent use cases at the cost of a higher ATL.

For instance, with a chunk length of 864, resulting in a transaction size of 489 KiB, the average throughput is approximately 23 TPS. This configuration can effectively serve 23 concurrent Structural Health Monitoring (SHM) applications while maintaining an average latency of less than 1.5 seconds.

V. Conclusions

This study showcases an innovative methodology for evaluating the impact of real-world factors on smart contract performance in industrial settings, consisting in a four-step process. Our approach leverages Average Transaction Latency (ATL) and Average Transaction Throughput (ATT) as key performance indicators for measuring and comparing outcomes across different blockchain technologies.

Applying this methodology to a Structural Health Monitoring (SHM) case study, we were able to assess its feasibility under a non-ideal deployment. Moreover, we leverage the obtained insights to identify an optimal configuration that offers the lowest ATL while satisfying the application's requirements.

In addition, this performance evaluation allows us to estimate the maximum number of industrial applications that can be served concurrently by the target smart contract deployment in real-world conditions. Importantly, we also show how to tailor the system to serve additional use cases at the expense of increased latency.

REFERENCES

[1] N. Elia *et al.*, "Smart Contracts for Certified and Sustainable Safety-Critical Continuous Monitoring Applications," in *Advances in Databases and Information Systems*. Springer, Aug. 2022, pp. 377–391.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, 2008.

[3] F. A. Alabdulwahhab, "Web 3.0: The Decentralized Web Blockchain networks and Protocol Innovation," in *2018 1st International Conference on Computer Applications & Information Security (ICCAIS)*. IEEE, pp. 04–06.

[4] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, no. 37, pp. 2–1, 2014.

[5] M. L. Shooman, *Reliability of computer systems and networks: fault tolerance, analysis, and design*. John Wiley & Sons, 2003.

[6] E. Dubrova, *Fault-Tolerant Design*. New York, NY, USA: Springer, 2013.

[7] D. Ford *et al.*, "Availability in globally distributed storage systems," in *OSDI'10: Proceedings of the 9th USENIX conference on Operating systems design and implementation*. USENIX Association, Oct. 2010, pp. 61–74.

[8] A. A. Tamimi, R. Dawood, and L. Sadaqa, "Disaster recovery techniques in cloud computing," in *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, 2019, pp. 845–850.